OS Homework 1

電資學士班 0710006 盧可瑜

2020 Fall, Introduction to Operating System

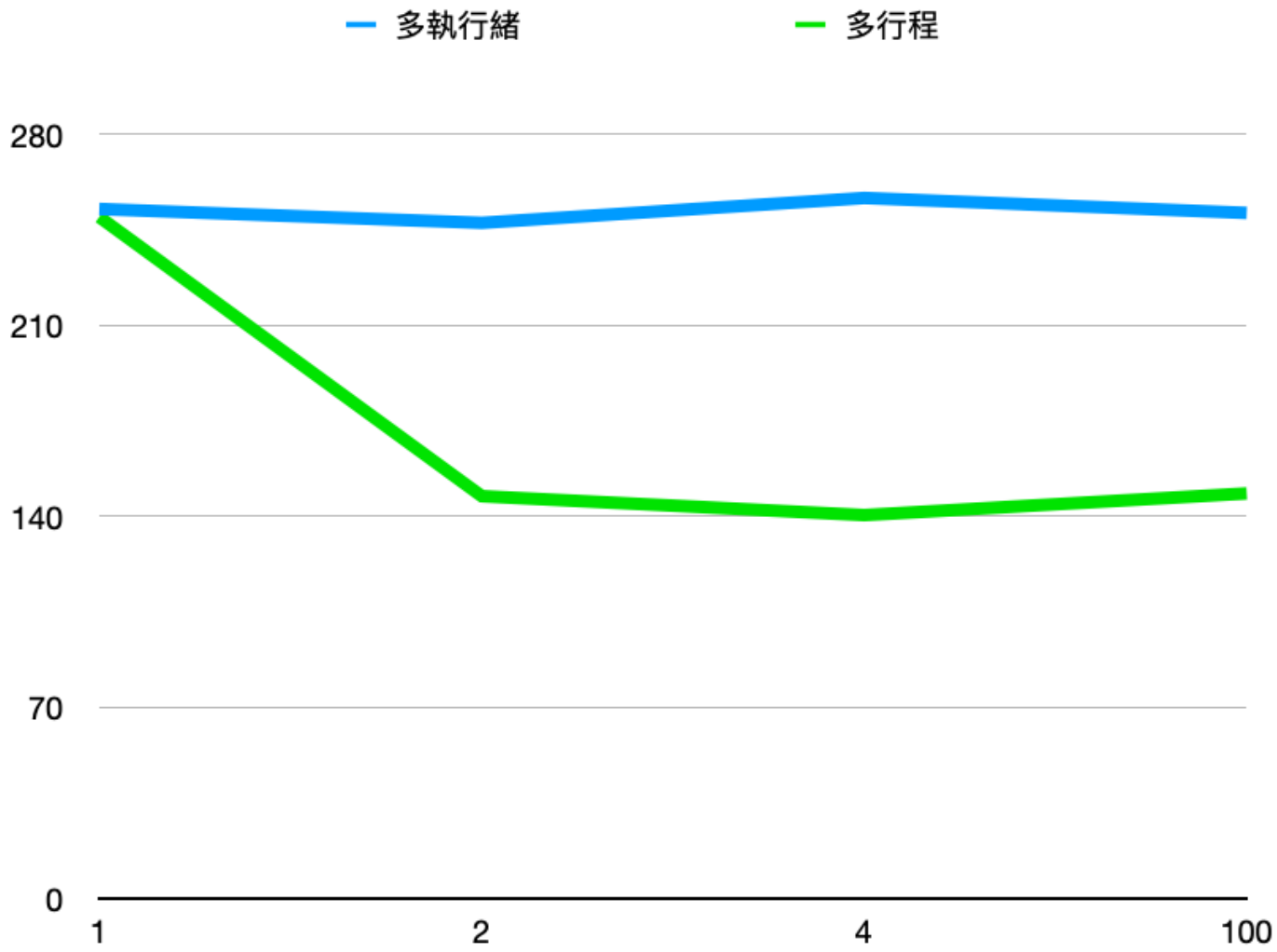
Date: 2021/1/3

Result

- 採用範例輸入資料 100 筆做測試。
- 由於輸入時間會影響結果，因此只計算讀完輸入資料後的執行時間。

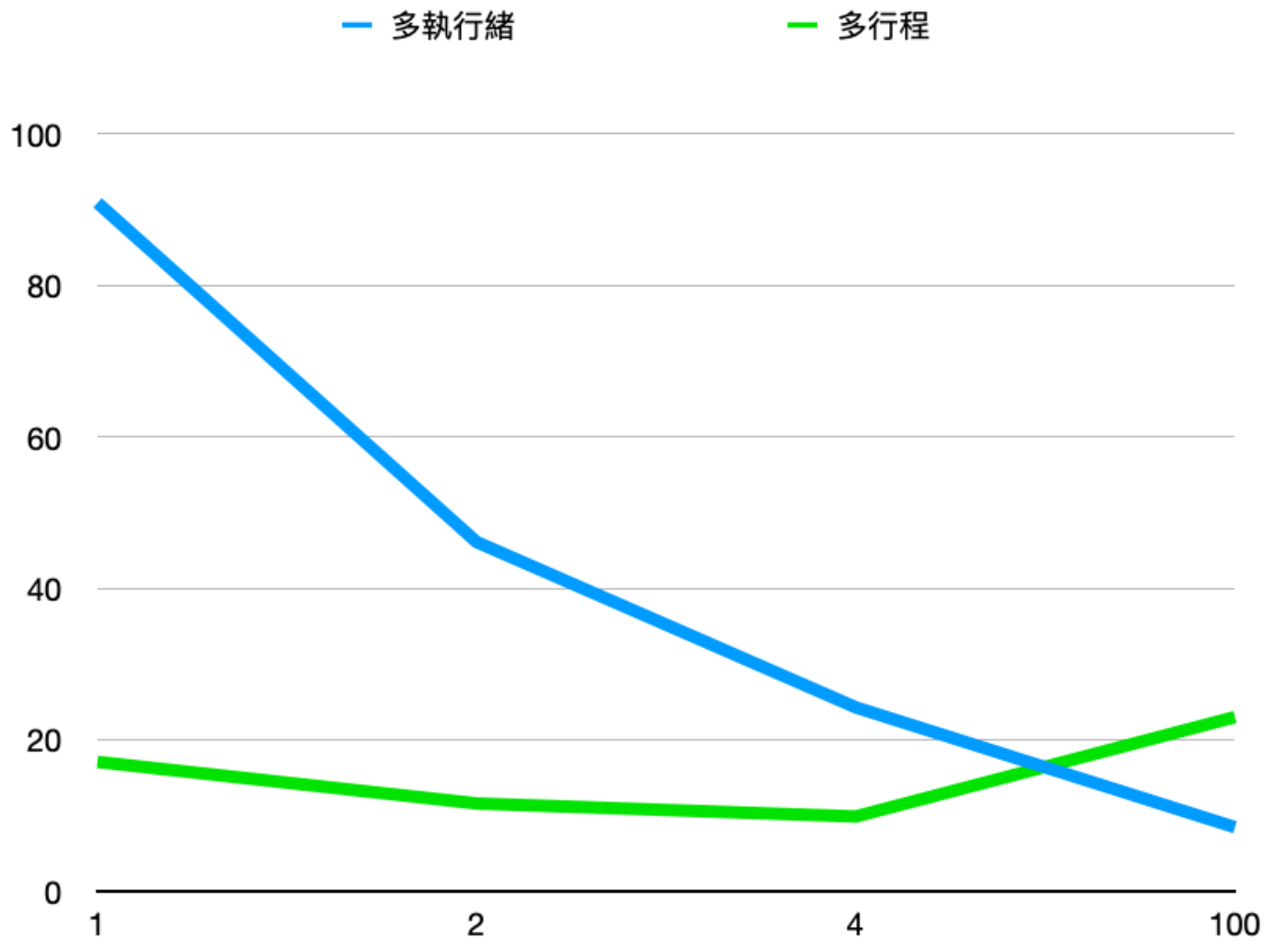
Task 1

方式	執行緒數/行程數	時間 (s)
多執行緒	1	252.7
	2	247.6
	4	256.7
	100	251.3
多行程	1	249.8
	2	147.4
	4	140.5
	100	148.4
協程		263.7



Task 2

方式	執行緒數/行程數	時間 (s)
多執行緒	1	90.9
	2	46.1
	4	24.3
	100	8.4
多行程	1	17.1
	2	11.6
	4	9.9
	100	23.0
協程		63.0



Conclusion

1. 多執行緒 Multithread

程序名稱	% CPU	CPU 時間	執行緒	閒置喚醒	PID
Python	98.6	9.08	5	481	26633

- 增加執行緒數量時，執行 Task2 比 Task1 具有速度上的影響。
- python 是直譯式語言，因此無法做到真正的 Multithread。但是他在爬網頁時具有優勢，因為需要一直等待網路回傳資料，所以 Thread 會經常閒置，這時用 Multithread 互相切換，就能利用那些等候的時間。

2. 多行程 Multiprocessing

程序名稱	% CPU	CPU 時間	執行緒	閒置喚醒	PID
Python	97.4	41.87	1	0	26626
Python	97.4	41.83	1	0	26627

- 可以發現在 Task 2 中，多行程並未有效改變效率，在 Task 1 中則有較明顯差異。可以得到結論，在運算量大的時候，使用 Multiprocessing 會得到優勢。
- 因為 Multiprocessing 是受到 CPU 數量影響，所以說當行程數設為 100 時並不會有任何優勢。而行程數從 1 到 2 時，可發現速度大約為兩倍。

3. 多行程、多執行緒、協程 (Coroutine)

- Coroutine 是指程序執行中可被中斷，他在 Task1 這種密集運算的情況下，沒有任何的優勢。而在 Task2 中，則具有類似 Multithread 的優勢。
- Coroutine 和 Multithread 的差別在於，我們可在使用 Coroutine 時，自行決定切換的時機點，而 Multithread 則是透過 OS 排程。
- 在 Task2 中 numProcess = 1 比 numThread 快，這可能跟 python 中模組的實踐有關，可能測試數量不夠多，無法妥善比較。
- 重新以 Task2 的資料複製為 4 份（400 筆）做測試，發現兩者差異就不像原本差了 5 倍，因此當資料夠多時，應該就不會有太大的差異。

Method	numProcess / numThread	Time(s)
Multithread	1	202.9
Multiprosess	1	151.9

Detail of Implementation

1. Environment

- `python 3.8.3`

2. `PoW()`

- 使用 `hashlib` 的 `sha256()`。
- 採用窮舉的方法，方便比較速度。
- 除此之外可考慮使用隨機的方法生成字串。

3. Retrieve Website Title

- 使用 `BeautifulSoup`。

4. Multithreading

- 將任務平均分配給每個 Thread。
- 因為 Task2 每個任務的執行時間可能差異很大，所以可試著將輸入 Shuffle 後再丟入 Threads，如此多實驗幾次後平均得到的結果可能會更精確。

4. Multithreading

- 使用 `multiprocessing.Pool`。

Code

```python
import time
import itertools
import hashlib
import requests
from bs4 import BeautifulSoup
from multiprocessing import Pool
import threading
import asyncio

query = []
ch = [chr(i) for i in range(33, 127)]

def getInput():
    workload = int(input(""))
    L = [int(x) for x in input("").split()] + [0]
    return workload, L[0], L[1], int(input(""))

def WebTitle(url):
    r = requests.get(url)
```

```python
    while r.status_code != requests.codes.ok:
        r = requests.get(url)
    soup = BeautifulSoup(r.text, 'html.parser')
    print(soup.find('title').text)

def checkSHA(s):
    sha = hashlib.sha256()
    sha.update(s.encode('utf-8'))
    code = sha.hexdigest()
    return code[0:5] == '00000'

def PoW(s):
    for i1, i2, i3, i4, i5 in itertools.product(ch, ch, ch, ch ,ch):
        tmp = i1 + i2 + i3 + i4 + i5 + s
        if checkSHA(tmp):
            print(tmp)
            return

def work(L, R, func):
    for taskId in range(L, R):
        func(query[taskId])

async def coroutine(func, s):
    await loop.run_in_executor(None, func, s)

if __name__ == '__main__':
    workload, taskMethod, numMulti, numTask = getInput()
    for i in range(numTask):
        query.append(input(""))
    timeStart = time.time()
    func = PoW if workload == 1 else WebTitle
    if taskMethod == 1: # Multithread
        threads = []
        for i in range(numMulti):
            numThreadTask = (numTask // numMulti + 1)
            taskL = numThreadTask * i
            taskR = min(numTask, taskL + numThreadTask)
            threads.append(threading.Thread(target = work, args = (taskL, taskR, func,
            threads[i].start();
        for thd in threads:
            thd.join()
    elif taskMethod == 2: # Multiprocess
        with Pool(numMulti) as pool:
            pool.map(func, query)
            pool.close()
            pool.join()
    elif taskMethod == 3: # Coroutine
        tasks = []
```

```
68          loop = asyncio.get_event_loop()
69          for qry in query:
70              tasks.append(loop.create_task(coroutine(func, qry)))
71              loop.run_until_complete(asyncio.wait(tasks))
72      timeEnd = time.time()
73      print("Run time: {}.".format(timeEnd - timeStart))
```